Cutting the margin

By L. Clarke & A.R. Hill

These hints may help you shorten a line which is marginally too long to type into the 64 character input buffer (ie, exceeds two lines on the screen).

The word, ''PRINT'' may be entered as a question mark (?) saving four character spaces. The word, ''REM'' or '':REM'', may be replaced by an apostrophe ('), saving either two or three character spaces.

The computer will convert the (?) to the token for ''PRINT'' when it is stored in the memory, so that when the line is listed, it will appear as ''PRINT''. If the line then exceeds 64 characters on the screen, it will ''wrap around'' onto the next line, but will still function normally. Of course, the on screen editor uses the input buffer, and any attempt to edit a line exceeding 64 characters will result in the loss of all text after the 64th character displayed on the screen!

The following functions must be POKEd into an existing line in a BASIC program.

Example 1:

If the first line of a program is used (eg, line number 1), then the first memory location past the line number is 31469. This does not change regardless of the number of digits in the line number because all line numbers are stored in memory as a two byte code.

Example 2:

If you want use any of the following functions in the middle of a program — just type up to the place where you wish to insert the function, place a dummy character in that position, and press [RETURN].

Immediately (with no line number) type in the following

PRINT PEEK(30969) + 256 * PEEK(30970) −4.

This will give you the memory location of the last character you typed into the last program line (in this case the dummy character). Memorise this number (write it down!) then finish typing in the BASIC line, continuing immediately after the dummy character.

When you have finished typing in the line, LIST it and check it is correct, because once you have POKEd the function code into the memory location in which your dummy character is stored, you will not be able to edit that line!

You may now POKE the function code into the memorised location which holds the dummy character. If the memory address should exceed 32767, it is first necessary to subtract 65536 to reduce it to an integer for the POKE command to work.

It is assumed you have made no changes (insertions or deletions) to the program before the dummy character, because these would have changed its memory location.

Election graphics

The explicit graphics used on the television in the run-up to the recent general election, particularly in Eye Witness News, were produced on the Poly. Staff of the Auckland office of Progeni were contracted to work with producers and directors — with very little time between receiving data and completing the task, according to an item in the Polytalk newsletter.

Function	No	How to Use	Description
RANDOM	134	1# POKE31469,134	Makes RND( ) statement more random.
DEFINT	153	1#A,B POKE31469,153	Defines all variable starting with ''A'' or ''B'' as being integers.
DEFSNG	154	1#C,D POKE31469,154	Defines all variables starting with ''C'' or ''D'' as being single precision (6/7 digit floating).
DEFDBL	155	1#E,F POKE31469,155	Defines all variables starting with ''E'' or ''F'' as being double precision (16/17 digit floating).
ON	161	1# POKE31469,161	Used with ON GOTO, ON GOSUB or ON ERROR (see below).
ERROR	158	1#* POKE31469,161 POKE31470,158	Used as ''ON ERROR GOTO line no''.
RESUME	159	1# 1#100 1#100 NEXT POKE31469,159	After error, return to error point. After error, GOTO 100. After error, return to the line after the one producing the error.
DELETE	182	1#150-300 POKE31469,182	Deletes lines 150 to 300 inclusive. Both lines 150 & 300 must exist.
AUTO	183	1# POKE31479, 183:RUN	Automatically prints line numbers starting at 10, increment of 10.
AUTO	183	1#500, 20 POKE31469, 183:RUN POKE30945,175	Automatically prints line numbers starting at 500, increment of 20. AUTO will print any existing lines found. If the AUTO function was halted with [BREAK], it will now continue from that point.
VARPTR	192	1#(X) POKE31469,192	Used to locate the memory address of a variable.
STRING$	196	1PRINT#(12,45) POKE31470,196	Will print 12 asterisks ''*'' (maximum length of string = 256 characters).
MEM	200	1PRINT# POKE31470,200	Tells the amount of unused memory left.
FRE	218	1PRINT#(A) POKE31470,218	Tells the number of unused bytes left in memory.
FRE	218	1PRINT#(A$) POKE31470,218	Tells the number of unused bytes left in the reserved string space.
CINT	239	1#Z POKE31469,239	Removes all digits after the decimal point.
CSNG	240	1#Z POKE31469,240	Converts numeric variable from double to single precision.
CDBL	241	1#Z POKE31469,241	Converts numeric variable from single to double precision.
FIX	242	1A=#(N) POKE31471,242	Removes all digits to the right of the decimal point. Doesn't round down negative numbers.
ERL	194	1PRINT# POKE31470,194	Returns the line number from which program branched to error routine.
ERR	195	1PRINT# POKE31470,195	Returns a value related to the type of error which last occurred.

These functions may be performed either with or without a line number.
For TRON (Trace ON) just POKE 31003,175
For TROFF (Trace OFF) just POKE 31003,0
The audible ''beep'' produced when a key is pressed can be controlled.
For BEEP ON just POKE 30779,32
For BEEP OFF just POKE 30779,0